

## Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

### Listing of Claims:

1. (Presently Amended) A computer system that employs a plurality of threads of execution to perform a parallel-execution operation in which the threads identify tasks dynamically and in which the computer system comprises:
- A) a mechanism that associates a separate status-word field with each of the threads; and
  - B) a mechanism that operates the threads in a manner that each thread:
    - i) each thread executes a task-finding routine to find tasks previously identified dynamically and performs tasks thereby found, with its associated status-word field containing a value indicating it is active, until the task-finding routine finds no more tasks;
    - ii) when the task-finding routine executed in step (i) finds no more tasks, that thread sets the contents of its associated status-word field to a value indicating it is inactive;
    - iii) after completing step (ii) and while the status-word field associated with any other thread contains a value indicating that the other thread is active, that thread searches continues to search for a task using the task-finding routine, and, if it finds one, sets its associated status-word field contents to a value indicating that it is active before attempting to execute a found task; and
    - iv) if during step (iii) when none of the status-word fields associated with other threads contains a value indicating that an associated thread is active and no task has been found, that thread terminates its performance of the parallel-execution operation.

1 2. (Original) A computer system as defined in claim 1 wherein the parallel-execution  
2 operation is a garbage-collection operation.

1 3. (Currently Amended) A computer system as defined in claim 1 wherein:  
2 A) each thread has associated with it a respective work queue in  
3 which it places task identifiers of tasks that thread identifies dynamically;  
4 B) the task-finding routine executed by ~~an-executing~~ that thread  
5 includes performing an initial search for a task identifiers in the work queue  
6 associated with ~~the-executing~~ that thread and, if that work queue contains no  
7 task identifiers that ~~the-executing~~ thread can claim, thereafter performing a  
8 further search for a task identifier in at least one other task-storage location.

1 4. (Original) A computer system as defined in claim 3 wherein the parallel-execution  
2 operation is a garbage-collection operation.

1 5. (Original) A computer system as defined in claim 3 wherein the at least one other  
2 task-storage location includes at least one work queue associated with a thread  
3 other than the executing thread.

1 6. (Original) A computer system as defined in claim 5 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over flow list.

- 1 7. (Original) A computer system as defined in claim 5 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 8. (Original) A computer system as defined in claim 5 wherein the further search  
2 includes repeatedly searching a work queue associated with a thread other than  
3 the executing thread until the executing thread thereby finds a task or has  
4 performed a number of repetitions equal to a repetition limit greater than one.
- 1 9. (Original) A computer system as defined in claim 8 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 10. (Original) A computer system as defined in claim 3 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over flow list.
- 1 11. (Presently Amended) A computer system as defined in claim 1 wherein the  
2 contents of all of the status-word fields, when taken together, form a status word  
3 that fits fit in a memory location accessible in a single machine instruction.
- 1 12. (Original) A computer system as defined in claim 11 wherein the parallel-  
2 execution operation is a garbage-collection operation.

1 13. (Original) A computer system as defined in claim 11 wherein each status-word  
2 field is a single-bit field.

1 14. (Previously Presented) A computer system as defined in claim 13 wherein each  
2 single-bit field contains a logic one to indicate that the associated thread is active  
3 and contains a logic zero to indicate that the associated thread is inactive.

1 15. (Presently Amended) For employing a plurality of threads of execution to perform  
2 a parallel-execution operation in which the threads identify tasks dynamically, a  
3 method comprising:

4 A) associating a separate status-word field with each of the threads;  
5 and

6 B) operating the threads in a manner that each thread:

7 i) each thread executes a task-finding routine to find tasks  
8 previously identified dynamically and performs tasks thereby found, with  
9 its associated status-word field containing a value indicating it is active,  
10 until the task-finding routine finds no more tasks;

11 ii) when the task-finding routine executed in step (i) finds no  
12 more tasks, that thread sets the contents of its associated status-word  
13 field to a value indicating it is inactive;

14 iii) after completing step (ii) and while the status-word field  
15 associated with any other thread contains a value indicating that the other  
16 thread is active, that thread ~~searches~~ continues to search for a task using  
17 the task-finding routine, and, if it finds one, sets its associated status-word  
18 field contents to a value indicating that it is active before attempting to  
19 execute a found task; and

20 iv) if during step (iii) when none of the status-word fields  
21 associated with other threads contains a value indicating that an  
22 associated thread is active and no task has been found, that thread  
23 terminates its performance of the parallel-execution operation.

- 1 16. (Original) A method as defined in claim 15 wherein the parallel-execution  
2 operation is a garbage-collection operation.
- 1 17. (Currently Amended) A method as defined in claim 15 wherein:  
2 A) each thread has associated with it a respective work queue in  
3 which it places task identifiers of tasks that thread identifies dynamically;  
4 B) the task-finding routine executed by ~~an-executing~~ that thread  
5 includes performing an initial search for a task identifiers in the work queue  
6 associated with ~~the-executing~~ that thread and, if that work queue contains no  
7 task identifiers that ~~the-executing~~ thread can claim, thereafter performing a  
8 further search for a task identifier in at least one other task-storage location.
- 1 18. (Original) A method as defined in claim 17 wherein the parallel-execution  
2 operation is a garbage-collection operation.
- 1 19. (Original) A method as defined in claim 17 wherein the at least one other task-  
2 storage location includes at least one work queue associated with a thread other  
3 than the executing thread.
- 1 20. (Original) A method as defined in claim 19 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over-flow list.

- 1 21. (Original) A method as defined in claim 19 wherein the task-finding routine  
2 includes selecting in a random manner the at least one work queue associated  
3 with a thread other than the executing thread.
- 1 22. (Original) A method as defined in claim 19 wherein the further search includes  
2 repeatedly searching a work queue associated with a thread other than the  
3 executing thread until the executing thread thereby finds a task or has performed  
4 a number of repetitions equal to a repetition limit greater than one.
- 1 23. (Original) A method as defined in claim 22 wherein the task-finding routine  
2 includes selecting in a random manner the at least one work queue associated  
3 with a thread other than the executing thread.
- 1 24. (Original) A method as defined in claim 17 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over-flow list.
- 1 25. (Presently Amended) A method as defined in claim 15 wherein the contents of all  
2 of the status-word fields, when taken together, form a status word that fits fit in a  
3 memory location accessible in a single machine instruction.
- 1 26. (Original) A method as defined in claim 25 wherein the parallel-execution  
2 operation is a garbage-collection operation.

1 27. (Original) A method as defined in claim 25 wherein each status-word field is a  
2 single-bit field.

1 28. (Previously Presented) A method as defined in claim 27 wherein each single-bit  
2 field contains a logic one to indicate that the associated thread is active and  
3 contains a logic zero to indicate that the associated thread is inactive.

1 29. (Presently Amended) A storage medium containing instructions readable by a  
2 computer system to configure the computer system to employ a plurality of  
3 threads of execution to perform a parallel-execution operation in which the  
4 threads identify tasks dynamically and in which the computer system comprises:

5 A) a mechanism that associates a separate status-word field with each  
6 of the threads; and

7 B) a mechanism that operates the threads in a manner that each  
8 thread:

9 i) each thread executes a task-finding routine to find tasks  
10 previously identified dynamically and performs tasks thereby found, with  
11 its associated status-word field containing a value indicating it is active,  
12 until the task-finding routine finds no more tasks;

13 ii) when the task-finding routine executed in step (i) finds no  
14 more tasks, that thread sets the contents of its associated status-word  
15 field to a value indicating it is inactive;

16 iii) after completing step (ii) and while the status-word field  
17 associated with any other thread contains a value indicating that the other  
18 thread is active, that thread searches continues to search for a task using  
19 the task-finding routine, and, if it finds one, sets its associated status-word  
20 field contents to a value indicating that it is active before attempting to  
21 execute a found task; and

22 iv) if none of the status-word fields associated with other  
23 threads contains a value indicating that an associated thread is active,  
24 terminates its performance of the parallel-execution operation.

- 1 30. (Original) A storage medium as defined in claim 29 wherein the parallel-  
2 execution operation is a garbage-collection operation.
- 1 31. (Currently Amended) A storage medium as defined in claim 29 wherein:  
2 A) each thread has associated with it a respective work queue in  
3 which it places task identifiers of tasks that thread identifies dynamically;  
4 B) the task-finding routine executed by ~~an executing~~ that thread  
5 includes performing an initial search for a task identifiers in the work queue  
6 associated with ~~the executing~~ that thread and, if that work queue contains no  
7 task identifiers that ~~the executing~~ thread can claim, thereafter performing a  
8 further search for a task identifier in at least one other task-storage location.
- 1 32. (Original) A storage medium as defined in claim 31 wherein the parallel-  
2 execution operation is a garbage-collection operation.
- 1 33. (Original) A storage medium as defined in claim 31 wherein the at least one other  
2 task-storage location includes at least one work queue associated with a thread  
3 other than the executing thread.
- 1 34. (Original) A storage medium as defined in claim 33 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over flow list.



- 1 35. (Original) A storage medium as defined in claim 33 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 36. (Original) A storage medium as defined in claim 33 wherein the further search  
2 includes repeatedly searching a work queue associated with a thread other than  
3 the executing thread until the executing thread thereby finds a task or has  
4 performed a number of repetitions equal to a repetition limit greater than one.
- 1 37. (Original) A storage medium as defined in claim 36 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 38. (Original) A storage medium as defined in claim 31 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the work queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over flow list.
- 1 39. (Presently Amended) A storage medium as defined in claim 29 wherein the  
2 contents of all of the status-word fields, when taken together, form a status word  
3 that fits fit in a memory location accessible in a single machine instruction.
- 1 40. (Original) A storage medium as defined in claim 39 wherein the parallel-  
2 execution operation is a garbage-collection operation.

1 41. (Original) A storage medium as defined in claim 39 wherein each status-word  
2 field is a single-bit field.

1 42. (Previously Presented) A storage medium as defined in claim 41 wherein each  
2 single-bit field contains a logic one to indicate that the associated thread is active  
3 and contains a logic zero to indicate that the associated thread is inactive.

1 43. (Presently Amended) A computer signal representing a sequence of instructions  
2 that, when executed by a computer system, configures the computer system to  
3 employ a plurality of threads of execution to perform a parallel-execution  
4 operation in which the threads identify tasks dynamically and in which the  
5 computer system comprises:

6 A) a mechanism that associates a separate status-word field with each  
7 of the threads; and

8 B) a mechanism that operates the threads in a manner that each  
9 thread:

10 i) each thread executes a task-finding routine to find tasks  
11 previously identified dynamically and performs tasks thereby found, with  
12 its associated status-word field containing a value indicating it is active,  
13 until the task-finding routine finds no more tasks;

14 ii) when the task-finding routine executed in step (i) finds no  
15 more tasks, that thread sets the contents of its associated status-word  
16 field to a value indicating it is inactive;

17 iii) after completing step (ii) and while the status-word field  
18 associated with any other thread contains a value indicating that the  
19 associated thread is active, that thread searches continues to search for a  
20 task using the task-finding routine, and, if it finds one, sets its associated  
21 status-word field contents to a value indicating that it is active before  
22 attempting to execute a found task; and

23 iv) if during step (iii) when none of the status-word fields  
24 contains a value indicating that an associated thread is active and no task

25                    has been found, that thread terminates its performance of the parallel-  
26                    execution operation.

1    44.    (Original) A computer signal as defined in claim 43 wherein the parallel-execution  
2                    operation is a garbage-collection operation.

1    45.    (Currently Amended) A computer signal as defined in claim 43 wherein:  
2                    A)        each thread has associated with it a respective work queue in  
3                    which it places task identifiers of tasks that thread identifies dynamically;  
4                    B)        the task-finding routine executed by ~~an-executing~~ that thread  
5                    includes performing an initial search for a task identifiers in the work queue  
6                    associated with ~~the-executing~~ that thread and, if that work queue contains no  
7                    task identifiers that ~~the-executing~~ thread can claim, thereafter performing a  
8                    further search for a task identifier in at least one other task-storage location.

1    46.    (Original) A computer signal as defined in claim 45 wherein the parallel-execution  
2                    operation is a garbage-collection operation.

1    47.    (Original) A computer signal as defined in claim 45 wherein the at least one other  
2                    task-storage location includes at least one work queue associated with a thread  
3                    other than the executing thread.

1    48.    (Original) A computer signal as defined in claim 47 wherein:  
2                    A)        there is a size limit associated with each work queue;  
3                    B)        when a given thread dynamically identifies a given task that would  
4                    cause the number of task entries in the work queue associated with the given  
5                    thread to exceed the size limit if a task identifier that identities it were placed in  
6                    that work queue, the given thread instead places that task identifier in an  
7                    overflow list instead of in that work queue; and  
8                    C)        the at least one other task-storage location includes at least one  
9                    such over flow list.

- 1 49. (Original) A computer signal as defined in claim 47 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 50. (Original) A computer signal as defined in claim 47 wherein the further search  
2 includes repeatedly searching a work queue associated with a thread other than  
3 the executing thread until the executing thread thereby finds a task or has  
4 performed a number of repetitions equal to a repetition limit greater than one.
- 1 51. (Original) A computer signal as defined in claim 50 wherein the task-finding  
2 routine includes selecting in a random manner the at least one work queue  
3 associated with a thread other than the executing thread.
- 1 52. (Original) A computer signal as defined in claim 45 wherein:  
2 A) there is a size limit associated with each work queue;  
3 B) when a given thread dynamically identifies a given task that would  
4 cause the number of task entries in the word queue associated with the given  
5 thread to exceed the size limit if a task identifier that identifies it were placed in  
6 that work queue, the given thread instead places that task identifier in an  
7 overflow list instead of in that work queue; and  
8 C) the at least one other task-storage location includes at least one  
9 such over flow list.
- 1 53. (Presently Amended) A computer signal as defined in claim 43 wherein the  
2 contents of all of the status-word fields, ~~taken together, form a status word that~~  
3 ~~fits~~ fit in a memory location accessible in a single machine instruction.
- 1 54. (Original) A computer signal as defined in claim 53 wherein the parallel-execution  
2 operation is a garbage-collection operation.

- 1 55. (Original) A computer signal as defined in claim 53 wherein each status-word  
2 field is a single-bit field.
- 1 56. (Previously Presented) A computer signal as defined in claim 55 wherein each  
2 single-bit field contains a logic one to indicate that the associated thread is active  
3 and contains a logic zero to indicate that the associated thread is inactive.
57. (Presently Amended) A computer system that employs a plurality of threads of  
execution to perform a parallel-execution operation in which the threads identify  
tasks dynamically, the computer system including:
- A) means for associating a separate status-word field with each of the  
threads; and
  - B) means for operating the threads in a manner that each thread:
    - i) each thread executes a task-finding routine to find tasks  
previously identified dynamically and performs tasks thereby found, with  
its associated status-word field containing a value indicating it is active,  
until the task-finding routine finds no more tasks;
    - ii) when the task-finding routine executed in step (i) finds no  
more tasks, that thread sets the contents of its associated status-word  
field to a value indicating it is inactive;
    - iii) after completing step (ii) and while the status-word field  
associated with any other thread contains a value indicating that the other  
thread is active, that thread searches continues to search for a task using  
the task-finding routine, and, if it finds one, sets the status-word field  
contents to the activity-indicating a value indicating that it is active before  
attempting to execute a found task; and
    - iv) if during step (iii) when none of the status-word fields  
associated with other threads contains a value indicating that an  
associated thread is active and no task has been found, that thread  
terminates its performance of the parallel-execution operation.